计算机图形学

Computer Graphics

张思容
zhangsirong@buaa.edu.cn

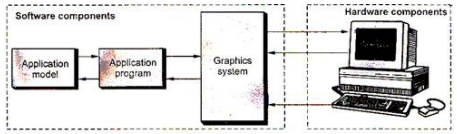
数学与系统科学学院，北京航空航天大学
School of Mathematics and Systems Science, Beihang University

November 1, 2012

Chapter 2: 图形学基础

1. 图形学的硬件与实现：图像系统
   - 直线扫描算法
   - 图形学硬件介绍:
   - 图形学软件介绍：openGL
2. 几何变换：仿射空间和射影空间
   - 仿射空间与向量几何
   - 仿射变换与射影变换
   - 射影空间和质点空间
3. 几何计算：裁剪与消隐***

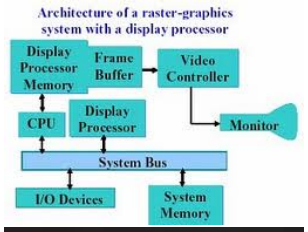
简单交互式图形系统



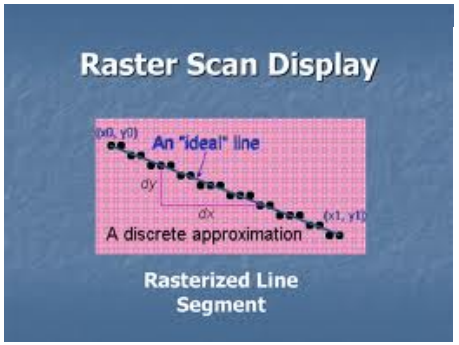
交互图形学的系统结构

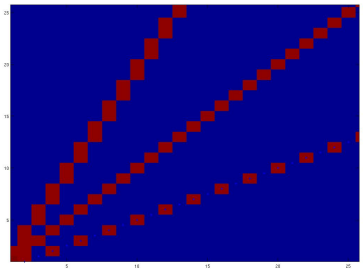
- 应用模型： 几何建模+物理模型（颜色与光）
- 应用程序：人机交互+投影变换；
- 图像系统： 扫描光栅显示；

光栅显示系统



例子：怎样画直线?





理想数学直线： 没有宽度，无限长!
MATLAB模拟例子：$y = x, y = 2x, y = x/2$

直线扫描算法I：DDA

一般直线段：$y = kx + b$, 给定$(x_0, y_0), (x_1, y_1)$.

- 模拟直线扫描：(x_0, y_0),计算$\delta(x), \delta(y)$,向量扫描算法
- 数字直线扫描：(x_0, y_0),计算$x++, y = k * x + b$,坐标取整, 离散扫描算法;
- DDA：数字微分扫描 (x_0, y_0),计算$x++, y+ = k$,坐标取整。增量扫描算法;
  注意：$|k| \leq 1$.
- 主要运算问题：浮点运算慢，有误差和坐标取整慢。

直线扫描算法II：中点算法和Bresenham算法

一般直线段：$y = kx + b$, 给定$(x_0, y_0), (x_1, y_1)$.
定义:水平线函数$F(x, y) = ax + by + c$，区域划分为$F > 0, F = 0, F < 0$
扫描算法的基本运算：找下一个点可以用$F(x + 1, y + 0.5)$做选择！
设$0 < k < 1, d = F(x + 1, y + 0.5), bd = F(x + 1, y + 1)$.

- 中点算法：$(x_0, y_0), d(0)$,p判断$d > 0, y = y; d \leq 0, y++$, 更新$d+ = \delta d$,循环。
  注：$d(0) = a + b/2$,可以用$2 * d$得到整数算法;
- Bresenham算法：$(x_0, y_0), bd(0)$,计算$bd > 0.5, y = y; bd \leq 0.5, y++$,更新$bd+ = \delta bd$,循环。
  注：可以定义$e = bd - 0.5, e(0) = -0.5$,可以用$2 * e * \delta(x)$得到整数算法;

Remark

中点算法和*Bresenham*算法对于直线和整数半径的圆扫描结果是一样的。

寻找更快的扫描算法？ 多步扫描和并行扫描.

圆扫描算法I：直接扫描***

一般方程：$y - y_c = \pm\sqrt{R^2 - (x - x_c^2}$.给定$(x_c, y_c), R$.

- 圆的8对称性：$(\pm x, \pm y), (\pm y, \pm x)$;
- 数字圆扫描：(x_0, y_0),计算$x++, y$,坐标取整, 离散扫描算法;
  像素不均匀！
- 参数圆扫描：(x_0, y_0),计算$\theta+ = \delta\theta, x, y$,坐标取整;
  三角函数慢！
- 主要运算问题：浮点运算慢，有误差和坐标取整慢。

圆扫描算法II：中点算法即Bresenham算法***

定义:水平线函数$F(x, y) = x^2 + y^2 - R^2$，区域划分为$F > 0, F = 0, F < 0$
扫描算法的基本运算：找下一个点可以用$F(x + 1, y - 0.5)$做选择！
设$0 < x < R/\sqrt{2}, d = F(x + 1, y - 0.5)$.设半径为整数，算法：

1. 给定园心和半径R，设$(x_0, y_0) = (0, R)$
2. 计算$d(0) = 5/4 - R$或$1 - R$;
3. $x++$, $d(k) < 0, y(k + 1) = y(k), d(k + 1) = d(k) + 2x(k + 1) + 1$;
   $d(k) > 0, y(k + 1) = y(k) + 1, d(k + 1) = d(k) + 2x(k + 1) + 1 - 2y(k + 1)$;
   特别：$x(k + 1) = x(k) + 1; y(k + 1) = y(k) - 1$;
4. 利用对称性得到其他七个点;
5. 循环直到$x \geq y$;

一维曲线的画图

曲线的类别

- 直线→多项式曲线
  特别： 二次曲线(conic section)
  $ax^2+by^2+cxy+dx+ey+f=0$
- 样条曲线： 若干多项式曲线段连接组成;
  特别：Bezier曲线，B样条曲线;
- 有理样条曲线（面) Nurbs,
- 任意参数曲线：（比如极坐标表示）;

画曲线方法： 利用直线逼近曲线:
找到若干点，直线连接;

- OpengGl： GL库函数仅有Bezier曲线;
  GLU有B样条，Nurbs(有理B样条)，球面，锥面等;
- MATLAB：参数表示;
  curvefit toolbox: B样条曲线;

二维图形的画图

二维图形的表示

- 规则图形：曲线（多边形)边界；区域表示（水平集）
- 不规则图形：区域表示（颜色？）
- 特殊图形： bitmap 块状图像；字符;

opengGL

- 矩形：*glRect*(),圆?
- 多边形：$glBegin(GL-PLOYGON)$; *glVertex*(); *glEnd*();
  $GL-TRIANGLES, GL-TRIANGLE-STRIP, GL-TRIANGLE-FAN, GL-QUAD, GL-QUAD-STRIP$
- 图像块：*glBitmap*(), *glDrawPixels*()
- 字符： *glutBitmapCharacter*(); *glutStrokeCharacter*()

多边形的扫描算法

基本的扫描算法：给定多边形按逆时针顺序的顶点集$v_i=(x_i,y_i)$，边为$e_i=v_iv_{i+1}$;

- 求交：计算每条扫描线和多边形每条边的交点
- 排序：对交点按x坐标排序;
- 配对：确定交点间线段是否再多边形内部;
- 画像素:

区域填充算法：给定多边形的边界的颜色和区域内一点（种子），假设区域是连通的,

- 连通性： 4连通，8连通;
- 递归算法：从种子出发，按连通性递归画像素;
- 边界填充boundary：边界颜色唯一确定;
- floodfill：内部颜色确定，用新颜色替换;

现实算法：扫描线+填充。

硬拷贝技术

重要术语:

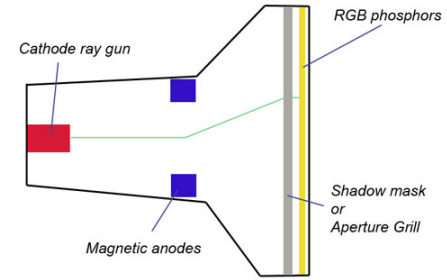
- 寻址能力addressability(每英寸的点个数 dpi), 点的尺寸,点间距离。
- DPI, 分辨率(resolution), PPI:
  分辨率小于DPI; 显示分辨率：VGA(640),XGA(1024),HDTV(1920);
  PPI: Windows 96 ppi,Apple: 72 ppi
- 彩色等级：bpp 1 bit:黑白； 8 bit: 256彩色;
  16 bit: 65536 彩色； 18bit: 26万； 24bit 真彩$(16,777,216)>1$千万。32bit?

常见设备:

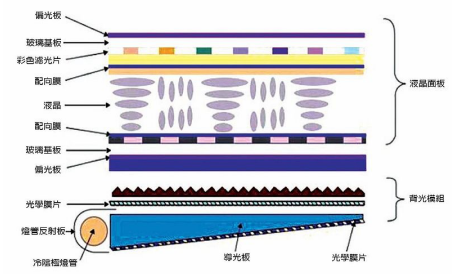
- 打印机：点阵；喷墨, 激光, 热传导;
- 绘图仪:
- 扫描仪，鼠标:

显示设备

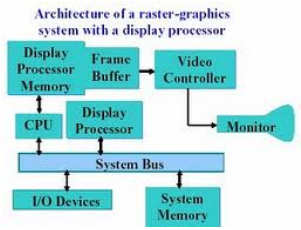
CRT:阴极射线管



LCD:液晶显示器(TFT: LED)



光栅显示系统



- CPU+视频控制器（video controller):
- 视频控制器：读帧内存(frame buffer) → 光栅扫描发生器 → 显示器
  常见扫描术语：交错或不交错60Hz； NTSC,PAL:
- 其他：视频查找表(彩色或灰度); 独立显示处理器GPU(扫描等硬件加速);

交互设备

- 定位设备： 输入板(手写),鼠标,跟踪球,游戏杆； 触摸板;
- 定值设备：键盘
- 选择设备：键盘+功能键;
- 输入设备：键盘或扫描仪;

3D设备?

图形学软件

常见软件包:

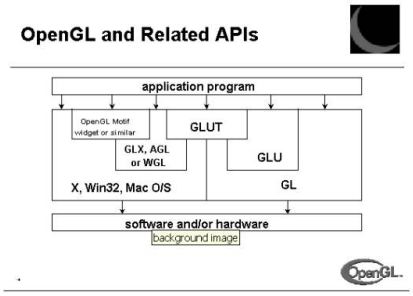
- 程序设计软件包: Openg GL, DirextX, VRML,Java2D,Java3D;
- 应用软件包： AutoCad, Flash, 3DsMax, Maya,...
- 其他通用软件: MATLAB, Mathematics,
  图像专用软件:

openg GL:

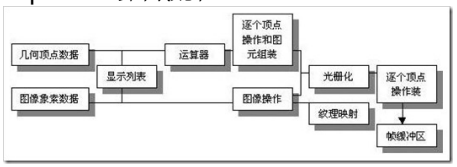
- 历史： 1992 1.0; 2.0,3.0
- 优点：OPEN： 工业标准； 跨平台, 通用,(编程规范,文档详细);
- 缺点：入门难, 图形学知识+数学知识;

OpenGL 的结构

API接口：



Pipeline:绘制流程



OpenGL 编程

- 组成：核心库(gl函数 115),应用库(glu函数 43),工具库(glut函数 30）辅助库(aux 31)；
  专用库：Windows: wgl函数，Linux: glX函数；
- 语法：C,C++,Fortran,Java 不同版本。
- 函数与变量：函数名前缀：gl,glu,glut等；
  常量名前缀：*GL*：
  变量类型：*GLint*等；参数调用：支持不同类型参数；
- 编辑器：Windows *VC* + +,linux：任何编辑器如gedit；
  头文件 *stdio.h*, *stdlib.h*, *math.h*
  openGL: *GL*/*gl.h*, *GL*/*glu.h* 或*GL*/*glut.h*.
- 编译： $gcc * .c - o * * * - lglut - lGL - lGLU - lm$

openg GL的运行机制：状态机制!!!
需要初始化glColor；需要打开或关闭某些功能(glEable);

简单openGL例子

openg GL的程序基本结构：

- 初始化：设置背景颜色；打开(光照，纹理);
- 显示设置：窗口，相机，变换；
- 建立模型(点，线,面);
- 主程序驱动：显示与交互；

代码：

```
int main(intargc, char **argv) {
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowSize(250, 250);
glutInitWindowPosition(100, 100);
glutCreateWindow("hello");
init();
glutDisplayFunc(display);
glutMainLoop();
return0; /* ANSICrequiresmaintoreturnint. */ }
```


仿射空间

点与向量的仿射空间是计算机图形学的基本对象；

- 向量与向量空间:向量的线性组合得到向量空间 $V : v, w, ...$
- 点与仿射空间：点的线性组合?
  点的仿射组合：$\sum_{k=0}^{n} a_k P_k = P_0 + \sum_{k=1}^{n} a_k (P_k - P_0)$,仅当$\sum a_k = 1, 0$有意义！
  仿射组合是封闭的，组成仿射空间。
- 点与向量的关系：$v = P - Q, P + v = Q$
  注意：$P + Q =?a * P =?$
- 向量的几何运算：长度，面积，体积.
  点积：$u \cdot v = |u||v| \cos(\theta)$
  叉积：$|u \times v| = |u||v| \sin(\theta)$
  行列式：$det(u, v, w) = (u \times v) \cdot w$

坐标表示与向量几何

坐标系：原点O,三个正交向量i, j, k

- 向量与向量空间:$V =< i, j, k >$的性组合；$v = xi + yj + zk \to (x, y, z)$
- 点与仿射空间：$P = O + (P - O) = O + (x, y, z) \sim (x, y, z)$
- 点与向量的运算：向量线性运算；
  注意：$v = P - Q, Q = P + v$

向量几何：

- 向量的几何运算：长度，面积，体积.
  点积：$u \cdot v = |u||v|\cos(\theta) = x_1x_2 + y_1y_2 + z_1z_2$
  叉积：$|u \times v| = |u||v|\sin(\theta)$,$det(ijk; x_1y_1z_1; x_2y_2z_2)$.
  行列式：$det(u, v, w) = det(x_1x_2x_3; y_1y_2y_3; z_1z_2z_3$
- 其他：余弦定理，正弦定理。
  面积：多边形Newell公式
  $area = 1/2\sum_{k=1}^{n}(P_k - Q) \times (P_{k+1} - Q)$,特别$Q = O$.
  点，线，面等的距离，

3D图形学中的变换

模型变换：共形变换，仿射变换： $A(p + v) = A(p) + A(v)$
投影变换：正交投影和透视投影。
刚体运动：正交变换；

- 平移：$P = P + w$
- 旋转：$P = Q + Rotate * (P - Q)$
  过Q点，沿方向u，旋转θ的变换矩阵：Rodrigues 公式：
  $v = (\cos\theta v + (1 - \cos\theta)(dot(u, v)u + \sin\theta cross(u, v)$
- 镜像：过Q点，法向量为N的平面做镜像；
  $v = v - 2dot(v, N)N$

伸缩变换和投影变换

伸缩变换：

- 均匀伸缩:到点Q的距离伸缩s;
  $P = Q + s(P - Q)$
- 非均匀伸缩:到点Q的距离沿方向单位向量w伸缩s;
  $P = P + (s - 1)dot(P - Q, w)w$

投影变换：

- 正交投影：到过点Q,法向量为N的平面正交投影;
  $P = P - dot(P - Q, N)N$
- 透视投影：从点E到点Q,法向量为N的平面的透视投影;
  $P = P + c(E - P)$
  $c = \frac{dot(Q-P,N)}{dot(E-P,N)}$
  $P = \frac{dot(E-Q,N)P+dot(Q-P,N)E}{dot(E-P,N)}$.

仿射变换的矩阵表示

坐标与矩阵

- 仿射空间:$v = (a, b, c)(i, j, k)^\tau$; $P = O + (x, y, z)(i, j, k)^\tau$;
- 仿射变换：A由$A(O), A(i), A(j), A(k)$完全决定；
  矩阵形式：$A = [A(i), A(j), A(k), A(O)]$是$3 \times 4$矩阵；
- 仿射坐标:$P = [P, 1], v = [v, 0]$
  $A = [A; 0, 0, 0, 1]$

仿射矩阵

- 仿射矩阵=线性变换+平移: $A = [L_A, w_A; 0, 1]$
- 线性变换: 基$u, v, u \times v$
  $L_A(v\Psi = av + b(u \cdot v)u + cu \times v$
  可以写成 $L_A = aId + bu^\tau u + cu \times \_$
- 平移向量w_A: 假设仿射变换有一个不动点；
  $A(Q) = Q$;则 $w_A = Q * (I - L_A)$.

常见变换的矩阵表示

仿射变换

- 平移：$Trans(w) = [I, w; 0, 1]$
- 旋转：$Rot(Q, u, \theta) = [Rot(u, \theta), Q * (I - Rot(u, \theta)); 0, 1]$
- 镜像:$Mirror(Q, N) = [I - 2N * N^{\tau}, 2dot(Q, N) * N; 0, 1]$
- 伸缩：$Scale(Q, u, s) = [I + (s-1)u * u^{\tau}, (1-s)dot(Q, u) * U; 0, 1]$

投影变换

- 正交投影：$Ortho(Q, N) = [I - N * N^{\tau}, dot(Q, N) * N; 0, 1]$
- 透视投影:
  齐次坐标：$(x, y, z, w) = (x/w, y/w, z/w, 1)$;
  $P = \frac{dot(E-Q,N)P + dot(Q-P,N)E}{dot(E-P,N)}$.
  $Persp(E, Q, N) =$
  $[dot(E - Q, N) * I - E * N^{\tau}, dot(Q, N) * E; -N, dot(E, N)]$

3D图形学中的射影空间

矩阵计算的几何问题:
点和向量的几何空间？齐次坐标的几何解释?
透视变换作用在向量上?
射影空间:

- 射影空间：仿射点和无穷远点;
  点=齐次坐标等价类：$[x, y, z, w]$ vs$[x, y, z, 0]$
  齐次坐标：过一点的直线 vs 某一个无穷远方向;
- 射影变换：可用矩阵表示(保持等价类);
  仿射变换：仿射点到仿射点，无穷远到无穷远点;
  透视变换：不平行与投射平面的向量到仿射点； 一般情形：***
- 射影几何：Poincare 圆盘:
  每条直线相交(非欧几何之一);
  存在对偶定理，曲线的Bezout定理;

注：射影几何不符合直观；无向量；没有线性运算！

质点空间：通用模型

质点:(mP, m),m是质量，P是位置;
向量：$(v, 0)$是速度(动量);
质点空间:

- 线性运算：标量乘法(质量的大小)
  加法：质点加法：Archimedes 杠杆定理
  $(m_1P_1, m_1) + (m_2P_2, m_2) = (m_1P_1 + m_2P_2, m_1 + m_2)$
  向量加法（普通加法)
  向量加质点:$(mP, m) + (v, 0) = (m(P + v/m), m)$
- 质点空间是四维线性空间！齐次坐标有物理意义。
  点(带质量的仿射点);仿射点是单位质量；向量是零质量的点;
  向量空间是质点空间的子空间；(3维);
- 线性变换：任何4×4的矩阵皆可。
  仿射变换：向量到向量，仿射点到仿射点，不改变质量?
  透视变换: 质点到质点平面上质点；（可能有向量变为点);
- 质点几何：比较自然(欧氏几何),可以线性运算；包含射影变换！
  注：四维不易想象；需要深入理解;

3D图形学的透视和伪透视变换

质点空间的透视变换：视点E,投影点P,考察质点加法:
$P = (m_1E + m_2P)/(m_1 + m_2) = \frac{dot(E-Q,N)P + dot(Q-P,N)E}{dot(E-P,N)}$
伪透视变换：图形学需要保存深度信息用于画图！

- 观察场景：由视点和投影平面决定的视觉棱锥形;
  有限的视觉棱锥体pyramid of vision: 选择近平面和远平面;
  一般情形：$glFrustum(xwmin, xwmax, ywmin, ywmax, dnear, dfar)$
  对称:$gluPerspective(theta, aspect, dnear, dfar)$，视场角$\theta$,纵横比$h/w$
- 伪透视变换：将视觉棱锥形变成长方体(x, y, z),其中透视可用正交投影(x, y),z是深度向量用于画图决定;
  伪透视变换=透视投影+深度向量，$depth(P, S) = dot(Q - P, N)N$
- 伪深度:直接向量加法得到非线性变换!
  质点加法:$P = Persp(E, Q, N)P + depth(P, S) = (dot(E - Q, N)P + dot(Q - P, N)E, dot(E - P, N)) + (dot(Q - P, N) * N, 0)$
  伪深度:$pseudodepth(P, E, S) = \frac{dot(Q-P,N)}{dot(E-P,N)}$;
  注：伪深度保持相对深度，可用于画图决定隐藏面！

裁剪 clipping***

图形学中给定规范化坐标系，去掉位于坐标系外面的几何物体，即裁剪。

- 点的裁剪：直接坐标判断。
- 线段裁剪：可能部分可见；
- 区域裁剪：边界裁剪加上填充；
- 曲线裁剪： 类似直线，复杂判别方程；
- 字符裁剪：可以选择全部，或部分。

openGL中的clipping
缺省：规范化投射加裁剪：6个平面$x = \pm 1, y = \pm 1, z = \pm 1$.
可以指定任意平面$glClipPlane(id, paramaters)$; $glEnable(id)$; 参数为(a, b, c, d).
注：一般裁剪在投影平面上实现；
以上任意平面裁剪在观察坐标系中3D中实现，程序会变慢！

2D线段裁剪算法***

简单算法：直接求交，乘法除法多，容易bug。 通常增加测试，减少求交。
Cohen-Sutherland算法：

- 矩形窗口编码：九个区域；
- 每个线段端点判断；全内，全外，可能交；
- 可能交情形再求交。

梁友栋-Barsky 算法

- 建立线段参数化模型；
- 每个线段端点判断；全内，全外，相交；
- 仅仅相交情形再求交。

3D裁剪和消隐***

参见后面内容。